File Systems and Object Storage

CS 240 - The University of Illinois

Wade Fagen-Ulmschneider

November 2, 2021

File Systems

All modern systems utilize an Operating System to facilitate the storage of data in units called “files”:

```
ls -la   # Lists file in the current directory
```

```
$ ls -la
```

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020  may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021  mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021  mosaic.py
```

```
$ ls -la
```

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020 may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021 mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021 mosaic.py
```

Permission Bits [1]

[3] Link Count

User & Group Ownership [2]

[4] File Size (bytes)

Date Last Modified [5]

File Name [6]

[1]: Permission Bits

```
5 -rw-r--r--  [...]
6 -rwxrwxrwx  [...]
7 Drwxrwxrwx  [...]
```

[2]: File Owner and File Group

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
```

[3]: File System Links

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020  may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021  mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021  mosaic.py
```

[4]: File Size

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020  may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021  mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021  mosaic.py
```

[5]: Last Modified Date

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020  may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021  mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021  mosaic.py
```

[6]: File Name

```
1 drwxr-xr-x 5 waf  waf  4096 Oct 11 17:59 .
2 drwxr-xr-x 3 root root 4096 Aug 31 15:15 ..
3 -rw------- 1 waf  waf  9405 Oct 17 03:15 .bash_history
4 -rw-r--r-- 1 waf  waf  3771 Aug 31 15:15 .bashrc
5 -rw-r--r-- 1 waf  waf    65 Sep  9 16:01 .gitconfig
6 -rwxrwxrwx 1 waf  waf  5283 May  4  2020 may4.csv
7 drwxrwxrwx 1 waf  waf  4096 Apr  3  2021 mosaic
8 -rwxrwxrwx 1 waf  waf  6275 Mar  8  2021 mosaic.py
```

[6]: File Name

File Extensions:

“Dot Files”:

Why does local file storage not work on a cloud-scale system?

Object Storage Systems

Object Cloud Storage Systems

Instead of using local file storage, large data storage in the cloud-based systems are commonly stored as “objects”. These objects (files) are organized into _____.


CreateBucket

Amazon AWS S3 CreateBucket REST API

```
1  PUT / HTTP/1.1
2  Host: Bucket.s3.amazonaws.com
3  x-amz-acl: ACL
4  x-amz-grant-read: GrantRead : UserList
5  x-amz-grant-write: GrantWrite : UserList
6  x-amz-grant-full-control: GrantFullControl : UserList
7  x-amz-grant-read-acp: GrantReadACP : UserList
8  x-amz-grant-write-acp: GrantWriteACP : UserList
…  [...]
```

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

Bucket in Request

```
2 Host: Bucket.s3.amazonaws.com
```

Bucket: Name of the bucket. *[Required]*

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

ACL in Request

```
3 x-amz-acl: ACL
```

ACL: The canned Access Control to apply to the bucket.

Allowed Values:

private | public-read | public-read-write | authenticated-read

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

UserList in Request

```
4 x-amz-grant-read: GrantRead : UserList
```

UserList: You specify each grantee (user) as a type=value pair, where the type is one of the following:

id – if the value specified is the canonical user ID of an AWS account
uri – if you are granting permissions to a predefined group
emailAddress – if the value specified is the email address of an AWS account

Ex: `x-amz-grant-read: id="11112222333",id="444455556666"`

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

ACP vs non-ACP

```
4 x-amz-grant-read: GrantRead : UserList
5 x-amz-grant-write: GrantWrite : UserList
6 x-amz-grant-full-control: GrantFullControl : UserList
7 x-amz-grant-read-acp: GrantReadACP : UserList
8 x-amz-grant-write-acp: GrantWriteACP : UserList
```

x-amz-grant-read grants permission for the file itself; **x-amz-grant-read-acp** grants permissions for the access control policies.

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

Q: In what ways does this differ from file systems?


PutObject

Amazon AWS S3 PutObject REST API

```
1  PUT /Key HTTP/1.1
2  Host: Bucket.s3.amazonaws.com
3  x-amz-tagging: Tagging
4  x-amz-acl: ACL
5  x-amz-grant-full-control: GrantFullControl : UserList
6  x-amz-grant-read: GrantRead : UserList
7  x-amz-grant-read-acp: GrantReadACP : UserList
8  x-amz-grant-write-acp: GrantWriteACP : UserList
…  [...]
20 Content-Length: ContentLength
21
…  Body
```

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Key in Request

```
1 PUT /Key HTTP/1.1
```

Key: Object identifier (“file name”), must be unique per bucket. *[Required]*

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Bucket in Request

```
2 Host: Bucket.s3.amazonaws.com
```

Bucket: Name of the bucket. *[Required]*

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Permissions in Request

```
4 x-amz-acl: ACL
5 x-amz-grant-full-control: GrantFullControl : UserList
6 x-amz-grant-read: GrantRead : UserList
7 x-amz-grant-read-acp: GrantReadACP : UserList
8 x-amz-grant-write-acp: GrantWriteACP : UserList
```

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Tagging in Request

```
2 x-amz-tagging: Tagging
```

Tagging: A key-value pair of tags associated with a specific object.

Ex: `tag1=value1&tag2=value2`

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Body in Request

```
20 Content-Length: ContentLength
21
…  Body
```

Body: The contents of the object is sent as the payload of the HTTP packet.

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

Q: Is there a directory structure similar to traditional file systems?

Q: In both traditional file systems and S3, names must be unique. However, tagging allows for multiple files to have the same tag. What design possibilities does this open up for us?

Amazon’s AWS Offering:

Microsoft’s Azure Offering:

Google Cloud Platform Offering: